MESA Intro and Nova Lab

Ryan Connolly
Michigan State University

October 28, 2014

Introduction

This is a brief introductory tutorial written for Ed Brown's research group at Michigan State University on getting started with MESA from scratch, learning some of the basics, and getting a feel for working with accreting white dwarfs. It's designed for beginners who have little to no experience using MESA, but should *not* be treated as comprehensive. That is, I'll focus on white dwarf models and novae specifically, but you should feel free to explore other aspects of MESA as well!

This tutorial is written for MESA version 6794 or later, and assumes you setup MESA using the MESA SDK by Rich Townsend (found here). If you have never used MESA before, Section 1 of this tutorial will guide you through installation. If you already have a functional MESA installation that fits the aforementioned requirements, you can skip to Section 2.

As a warning, my own MESA experience is almost exclusively on Mac with a mix of terminal usage and browsing with the "Finder". If you're using Linux or remotely accessing another machine you'll probably require more terminal practice/experience than I'll discuss explicitly here (apologies!).

Please send all questions and comments to connol65@msu.edu!

Contents

1 Installation and Setup

If you have MESA version 6794 installed already, you can skip to Section 2.

The MESA Sourceforge page already has a walkthrough for installing and compiling MESA, which is now easier than ever.

http://mesa.sourceforge.net/prereqs.html

This page should guide you through installation for your Mac or Linux machine. You can optionally forgo the preliminary stuff and skip down to Section 1.5, "Install the prerequisites (MESA SDK)" (at the above link).

Once you've successfully installed MESA, it would be very useful to continue on and do as much of the Getting Started tutorial as you can.

http://mesa.sourceforge.net/starting.html

The above guide covers the absolute basics of getting started, creating a fresh working directory, familiarizing yourself with running MESA and interpreting the output, etc. I'll go over many of the same things in the later sections of this tutorial as well, but it never hurts to have more practice!

If you run into any issues with the installation process, or have trouble with any of your first runs, email me and I'll do my best to help troubleshoot.

2 Example Test Suite Case: `wd2`

The rest of the tutorial will assume the user is very new to MESA (experienced users bear with me). As mentioned in Section 1, I still recommend going through the Getting Started tutorial to learn about the absolute basics of MESA. However, I'll still explain most steps in (excessive) detail for newcomers, so if you're familiar with running MESA and have a preferred way of doing things feel free to skim and skip steps where appropriate.

As a mini-lab, we're going to briefly run one of the test suite cases to get a feel for working with white dwarf models in MESA. The test suite is a diverse batch of models that are used by the developer to ensure many of the various capabilities of MESA remain functional and consistent as the software is developed on a daily basis. The test suite problems can be found in the `mesa/star/test_suite` directory. We're going to borrow one of the test suite problems, `wd2`, which evolves a 1 $M_\odot$ accreting white dwarf (WD) through a nova outburst.

2.1 Preparing the project directory

It's good practice to avoid doing any work in the actual MESA directory, so you don't cause yourself any confusion or clobber your projects when updating MESA in the future! For example, I keep a folder called `MESA Work` in my `Documents` directory for all of my MESA projects (thrilling, I know). Pick a place to start storing your MESA stuff and navigate there in a terminal window. Once there, make a copy of the `wd2` test suite with the following command:

```
cp -r $MESA_DIR/star/test_suite/wd2 wd2
```

As a note: feel free to do as much of this manually by drag-and-dropping in the Finder (or equivalent file browser for your machine) if you're more comfortable that way.

> **IMPORTANT!** The above command will only work if you've properly defined `MESA_DIR` as an environment variable, which will also be crucial moving forward! If you have an issue with the previous step, revisit Section 1.7 of the MESA installation guide. If that doesn't solve the issue email me with the details.

From here on out, we're going to work exclusively in *our* copy of `wd2`. Once you've successfully made a copy in your own work folder, there are a few changes we need to make before we jump in and run it. All test suite

projects have MESA_DIR manually defined for development purposes. We need to comment out these lines in two places so that they don't overwrite your own MESA_DIR.

Navigate to your copy of wd2 and open up the inlist file. Near the top, in the &star_job section, you will see a statement that defines mesa_dir. Comment this out (with an exclamation point, in Fortran) so the entire line looks like this:

```
!mesa_dir = '../../..'
```

The second location is in the makefile, located in the make folder inside our wd2 directory. Open up makefile and comment out the MESA_DIR line just like before.

At this point you *could* run the simulation successfully. However, the numerical output won't be very helpful for us right now, so let's have MESA give us a few simple plots during our run. In wd2, open inlist_wd2 and scroll down to the &pgstar section at the bottom. Add the following three lines to the &pgstar section (anywhere should work).

```
TRho_Profile_win_flag = .true.
show_TRho_Profile_legend = .true.
HR_win_flag = .true.
```

Note: indenting (tabs or spaces) is only for organization in MESA, so you can put as much or as little space as you prefer before anything you add to inlists.

Finally, scroll back up toward the top to the &star_job section of the inlist. Find the statement with pgstar_flag and **remove** the exclamation point to uncomment it.

```
pgstar_flag = .true.
```

This tells MESA to plot for us, according to what we've specified in the &pgstar section. At this point, we're ready to move on and start the run.

2.2 Running and output

Open a terminal window and navigate to your wd2 work directory. Perform the following sequence of commands:

```
./clean
./mk
./rn
```

After the "make" command (`./mk`) the project will compile and you should see some messages from `gfortran`. This process usually takes a few seconds. Once you "run" (`./rn`), you will begin to see terminal output and, after a bit, two plot windows should appear (move the first plot to uncover the second). The early time steps should look something like Figure 1.

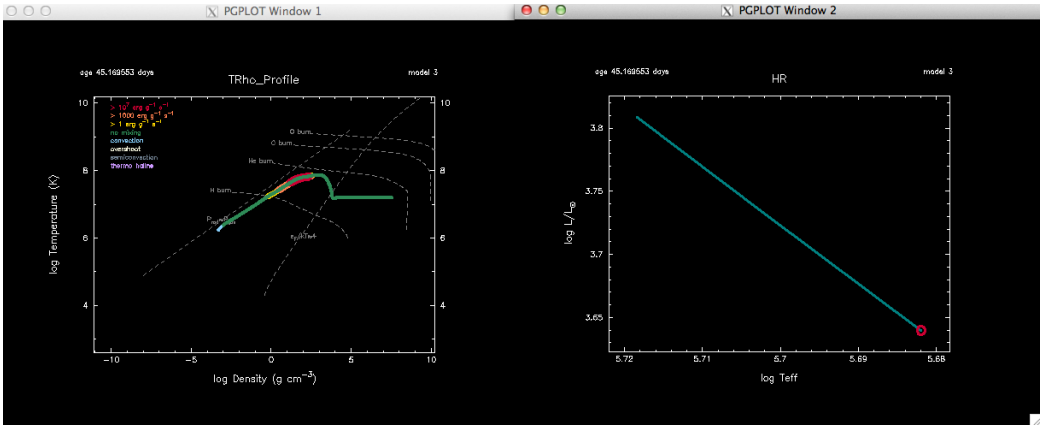

Figure 1: `TRho_Profile` and `HR` plots at the beginning of `wd2` run.

`TRho_Profile` is a log-log plot of temperature against density for the star. This profile is the "quintessential" MESA plot. It's almost always very handy and is included with multiple preset configurations in `pgstar`. The center of the star is to the right at high densities and the surface to the left. Labeled in gray are various lines marking things like conditions for different types of burning, degeneracy conditions, etc. The legend in the upper-left corner denotes what the various colors represent on the profile on the star. Our starting model is a white dwarf that has already accreted some mass and started hydrogen burning in the envelope. The degenerate core of the white dwarf is isothermal (hence the flat region to the right side of the plot) and the burning in the outer layers, denoted by the yellow/orange/red regions, has heated the accreted envelope of the star (accompanied by the increase in temperature we see around $\log \rho \sim 4$ if we trace outward from the core).

`HR` is, as you'd expect, an HR diagram plot of luminosity vs effective temperature, with hotter and brighter in the upper-left and cooler and dimmer in the lower-right. HR plots are also common with lots of `pgstar` presets. `TRho_Profile` and `HR` are great examples of each of the two main "brands" of plots `pgstar` can make: profiles and histories. Profiles, such as `TRho_Profile`, show various physical parameters or thermodynamic quantities (e.g. pressure, temperature, density, abundances, etc.) in each zone **throughout** the entire star **at a given time step**. History plots show the

evolution of properties of the whole star (e.g. central temperature, luminosity, mass) over **many time steps**. `HR` is an example of a history "track", where we're comparing two properties of the star as it evolves, but one can simply plot a variable against time or model number as well.

This test suite case starts with the white dwarf cooling after some previous outburst. As it cools and fades (moving down and to the right on the HR diagram), it's accreting mass at a constant rate, until it undergoes another outburst. Let the run continue and take some time to get a feel for each of the plots and what's happening as the outburst evolves. Watch the path that the star traces out on the HR diagram and see if you can find any correlations with what's happening in the T-ρ profile at various stages of the run. We'll go into more detail about what's physically happening in Section 3 when we build our own nova model with conditions more relevant to the project(s) at hand. The run should automatically end at model number 565, and the `pgstar` output toward the end should look like Figure 2.

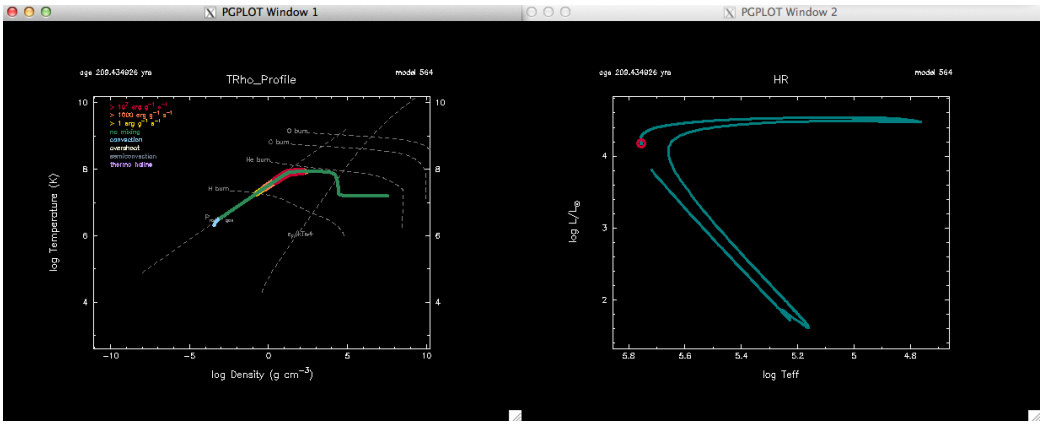

Figure 2: `TRho_Profile` and `HR` plots near the end of `wd2` run.

Depending on your machine, this run may take 10-15 minutes on average (maybe less, and hopefully not more!). Toward the end of the terminal output you should see a termination statement regarding `star_mass_max_limit`. This signifies that the run stopped as intended due to an imposed limit on the total mass of the star, which is increasing as it accretes mass. The details of why this stopping condition was chosen and other features of this particular model are unimportant for now. The intention of this section was simply to test your MESA installation (just in case), show how to borrow a test suite case as practice or a starting model, and give you a chance to stare at one example of a white dwarf undergoing a nova outburst, which we'll be seeing a lot of moving forward.

3 Building a Baseline Nova Model

3.1 Starting from scratch

3.2 Modifying reaction rates

4 Physics and Insight: What's Happening?

5 Suggested Further Reading

Below is a few additional useful tutorials and labs that I've compiled from the 2014 MESA Summer School and other sources.

Frank Timmes' `pgstar` plotting tutorial

Download the zip file from this link:

http://mesastar.org/teaching-materials/2014-mesa-summer-school-working-dir/timmes

Once extracted, you can find the lecture slides in the `presentations_pdf_files` folder. Ignore all the individual pdf's and just use `mesa_ss_2014.pdf`. The presentation/lab is nicely structured and starts off holding your hand with extremely detailed steps, but then leaves more and more up to the reader as the lab progresses. Perfect for learning `pgstar` as well as just general practice for experienced users. Keep in mind that when working with MESA in the future for your own research you can make one "master" `pgstar` inlist that has your favorite plots and easily copy it for multiple projects! After finishing the lab, it might be worthwhile investing some time creating your own plots with information relevant to your research using what you've just learned from the lab as a reference.

Kevin Moore's general MESA tutorial and `run_star_extras` lab

The lecture is here:

http://mesastar.org/teaching-materials/2014-mesa-summer-school-working-dir/moore

and the tutorial/lab itself (also linked in the lecture) is located here:

http://users.soe.ucsc.edu/~klmoore/index.html

Part 1 of the lab is a great intro even for relatively new MESA users. For someone with moderate MESA experience (months+), Parts 1 and 2 are both great practice as well as insightful. You may or may not be familiar with most of the things the first two parts cover, but the succinct and clean tutorial helped me wrap my head around a few things even after using MESA for over a year. Part 3 about adding physics hooks is more advanced and probably only necessary if you require it for your research.

Josiah Schwab's `run_star_extras` tutorial

Similar to Kevin Moore's tutorial above. Either of the following links should work.

https://github.com/jschwab/rse_tutorial

or

http://yoshiyahu.org/mesa2013.html

Again, just more good practice, and perhaps you'll get something out of it that you wouldn't in Kevin's, or vice versa.

Bill Wolf's `MesaScript` and `PyMesaReader`

An impromptu intro Bill threw together for and evening session at the 2014 MESA Summer School.

http://mesastar.org/teaching-materials/2014-mesa-summer-school-working-dir/evening-session-talks

(first entry, Evening_1_MesaScript_PyMesaReader.pdf)

`PyMesaReader` is probably more relevant. It's one of several tools floating around that helps get history and profile data output from MESA into Python easily and (relatively) intuitively. Everyone has different ways of doing this, and Python is flexible enough that each work pretty much as well as the next, but I know when I was new it could often be a pain and distraction, so it's nice to have tools/instructions on the ready.

`MesaScript` is a Ruby-based tool designed to let you use typical programming commands/variables/logical statements to make MESA inlists. For example, if you want to make a series of runs but want each star to have slightly different masses, `MesaScript` will allow you to output a bunch of organized inlists (e.g. "`inlist_1M`, `inlist_2M`, `inlist_4M`, `inlist_8M`, etc") with all of the same matching controls and commands except for the initial mass of the star. There are great examples in the PDF. On smaller scales this is something most of us have done at one point or another in our projects. While it's probably easier to do manually for a couple cases, if you wanted to run a large grid of simulations with dozens of inlists, `MesaScript` would help substantially in keeping things organized and (importantly!) ensuring all of your inlists match! When working with multiple inlists, it's easy to adjust controls to test something in only one place and forget that you did so down the road. Even if you aren't familiar with Ruby but have other Python/programming experience, it should be intuitive enough and the documentation and examples are good (and likely to be updated occasionally in the future as well).